MILLERS GRAPHICS

AUGUST 1984

$1.50

# THE SMART PROGRAMMER

I'm very sorry for the lengthy delay in getting this issue out. I really want to THANK everyone of you for your patience and understanding. We truly are trying and we have NOT lost interest in producing this publication. I do enjoy writing it.

As you can see we have not gone out of business nor do we have any such plans. Its interesting how these rumors get started. Its almost like someone wants us out of the 99/4A business. Well I have news for them its going to take a lot more than a couple of rumors to get us out. We like it too much! The past few months have been quite hectic with new product development and some time consuming legal dribble. In the new product department we have a lot of new goodies planned and started and as soon as they are ready we will let you know. In the not so new department we are working on finishing up that book and program we talked about last year.

The reason we have been sending back the renewals, that many of you have sent in, is because we have not completed your original subscription yet. I know the subscription form said "1 Year (12 Issues)" and that the first year is up. However we still have 5 more issues to go before the "12 Issues" are completed. You paid for 12 issues so we owe you 12 issues, I don't care how the rest of the publishing industry handles it. After these are completed we are looking at producing this publication in a different form. As you know it has been very hard to get this out in a timely fashion and still handle other new items. So what we are looking at is publishing a large issue 3 or 4 times a year and sending out a brochure to let you know its ready. I have a hard time sleeping at night, since I'm so far behind on this newsletter, knowing that you believed it would be published every month. As soon as we know exactly how we are going to handle the newsletter after the first 12 issues we will let you know. We are NOT abandoning the 4A or compatible computer, so long as you don't. It is still the BEST Home Computer around and there are a lot of GOOD things about to happen, for all of us, from a number of companies. So lets continue to prove the industry wrong and show them that there truly is life after death!?

---

Within the next couple of months we will be sending out a new catalog to everyone on our mailing list. This catalog will contain some new software and books as well as all of our other items. What we would also like to include in this catalog is a complete listing of the 99/4A and TI PC Users Groups from around the world. Along with the User Groups list we will also include a complete listing of all the dealers that are currently carrying our products.

In order to provide the most up to date listing of the Users Groups we ask that every Group that would like to be listed to please send us the following information: Group Name, Complete Address, Person to Contact, Phone Number, Total Group Membership, Library Size, Group BBS Number and the subscription rate for your newsletter. If your group prefers not to give out some of the above information we'll understand and we will place an N/A in that section of the listing. Even though our database already contains most of the Users Groups we will only publish the names that are sent in. So please help us spread the word to all Groups, there are a lot of 4A users out there looking for groups and newsletters.

# Q&A

Before we start the Q & A section I must ask once again to PLEASE not send self addressed stamped envelopes with your questions. I hate to see money go to waste but time does not permit me to respond except through the newsletter. I also feel this is the best way since we receive many similar questions and I think everyone would like to read them. OK enough of that so lets get started.

What is the latest information on the Sci Tech RAM Disk Card?

Well its not a Sci Tech project any more it has been turned over to another company by the name of Computronics which is in Corona California. They can be reached at (714) 369-5964. I've been told that this phone number is answered 24 hours a day.

This project is way behind schedule so I'm still not sure when it will be ready or how much the unit will cost. We were only contracted to do the software for the unit in two phases. Phase one of the software has been completed since October of 84', all it needs is a prototype card to marry it to for testing purposes. After it is tested out we can move on to phase two and complete the software. I believe that the target price for this unit with 128K of RAM will be comparable to the Foundation 128K Card. Since it is now being handled by a company that is new to the 4A market place please go easy on them. However, a phone call with your vote of confidence might help speed things up a little!.

---

Are the rumors about the 99/8 clone true? Do you think it is for real this time?

From what I currently know about this new computer it should be much better than a 99/8! So lets not call it a 99/8 clone any more lets just call it the the "Mystery Machine". I know we have all heard this story many times before but this time it really looks like we will have a new and very powerful big brother for our 4A's. The exact details of this computer are currently not available for release. I believe the company that is working on it is planning on debuting it at the 1985 June CES show in Chicago. So keep your fingers crossed and we'll keep you posted.

---

What the heck is going on between Millers Graphics and CorComp.

Because of time consuming legal dribble I can't say too much at this time. As you may be aware CorComp filled for Bankruptcy back in September of 84'. Well we were one of the companies that they filled against. Ouch! We thought we could work things out with them but alas communications have broken down. No Pay - No Work. Until the legal system gets through with this mess we can't say much more so stay tuned to this spot for the next episode of "As The Electron Turns".

P.S. Many thanks to everyone on Compuserve and The Source for your support on this matter.

---

LETS GET STARTED WITH THE FUN STUFF

The following XB program loads the uncompressed assembly file on the next page to activate an interrupt driven time clock on your monitor. After assembling the source code, type in and save this XB program as DSK1.LOAD. Then whenever you select XB from the menu the clock will start up. Have fun.

```
1 !    Clock Loader
  (DIS/FIX 80 obj file)
and clock setter program
   by Paul Schippnick

2 CALL INIT :: CALL LOAD("DS
K1.CLOCK"):: CALL LINK("CLOC
K")

3 DISPLAY AT(12,1):"TIME?___
___" :: ACCEPT AT(12,6)SIZE(
-6)BEEP VALIDATE(DIGIT):TIME
$ :: CALL LINK("SETCLK",TIME
$)

4 CALL CLEAR :: CALL LOAD(-3
1952,255,231,255,231)!
    This clears the load
 program out of memory like
        'NEW' does.
```

```
********************************************************************************

*      CLOCK ROUTINE FOR USE BY EXTENDED BASIC - ASSEMBLE THIS FILE UNCOMPRESSED

*           ROUTINE IS LOADED BY:
*                                      CALL INIT
*                                      CALL LOAD("DSK1.XCLOCK")
*                                      CALL LINK("CLOCK")

*           EXECUTION BEGINS UPON RETURN TO X-BASIC

*           TO SET CLOCK (ASSUME TIME IS 3:31 PM):
*           THE FORMAT IS HHMMSS

*                                      CALL LINK("SETCLK","153100")

********************************************************************************

* DOUGLAS C. WARREN
* 12/05/84

*===============================================================================

       DEF  CLOCK,SETCLK

VMBW   EQU  >2024         VDP MULTIPLE BYTE WRITE EQUATE
STRREF EQU  >2014         STRING REFERENCE EQUATE
GPLWS  EQU  >83E0         GPL WORKSPACE ADDRESS
STATUS EQU  >837C         STATUS BYTE ADDRESS
NEXT   EQU  >70           NEXT ENTRY IN GPL INTERPRETER
ISR    EQU  >83C4         ISR HOOK ADDRESS
TIMER  EQU  60            COUNTER FOR 60th OF SECOND LOOP COUNTER

COUNTR DATA TIMER         LOOP COUNTER
WSR    BSS  6             R0-R2! BEGINNING OF OUR WORKSPACE REGISTERS
R3LB   EQU  $+1
       DATA >8090         R3!  A SPACE AND THE TEN'S HOUR DIGIT
       DATA >909A         R4!  THE ONE'S HOUR DIGIT AND A COLON
R5LB   EQU  $+1
       DATA >9090         R5!  THE TEN'S AND ONE'S MINUTES DIGITS
R6LB   EQU  $+1
       DATA >9A90         R6!  A COLON AND THE TEN'S SECOND DIGIT
R7LB   EQU  $+1
       DATA >9092         R7!  THE ONE'S SECOND DIGIT AND COMPARISTON DATA
R8LB   EQU  $+1
       DATA >9096         R8!  COMPARISON DATA
R9LB   EQU  $+1
       DATA >9A94         R9!  COMPARISON DATA
       DATA >0100         R10!
R11LB  EQU  $+1
       DATA >6060         R11! DATA USED TO CONVERT ASCII DATA FOR BASIC SCREEN
R12LB  EQU  $+1
R13LB  EQU  $+3
R14LB  EQU  $+5
R15LB  EQU  $+7
       DATA >0006         R12! BEGINNING OF STRING BROUGHT IN THROUGH 'SETCLK'
       BSS  6             R13-R15! SPACE FOR DATA BROUGHT IN THROUGH 'SETCLK'

*===============================================================================
HOOK   DATA START        THIS IS THE START OF OUR PROGRAM
CLOCK  MOV  @HOOK,@ISR   LOAD THE ISR HOOK WITH THE START ADDRESS
START  LWPI WSR          LOAD OUR WORKSPACE REGISTERS
       DEC  @COUNTR      HAS A MINUTE PASSED YET?
       JNE  SCRN         NO!
       LI   R0,TIMER     PREPARE TO RELOAD COUNTER
       MOV  R0,@COUNTR   RELOAD COUNTR FOR NEXT MINUTE COUNT
       A    R10,R7       INCREMENT ONE'S SECOND DIGIT
       CB   R9,R7        SEE IF 10 SECONDS HAVE PASSED
       JGT  SCRN         NO!
       MOVB R8,R7        RESET ONE'S SECOND DIGIT TO ZERO
*------------------------------------------------------------------------
       INC  R6           INCREMENT TEN'S DIGIT
       CB   @R8LB,@R6LB  SEE IF 60 SECONDS HAVE PASSED
       JGT  SCRN         NO!
       MOVB R8,@R6LB     RESET TEN'S SECOND DIGIT
*------------------------------------------------------------------------
       INC  R5           INCREMENT ONE'S MINUTE DIGIT
       CB   R9,@R5LB     SEE IF 10 MINUTES HAVE PASSED
       JGT  SCRN         NO!
       MOVB R8,@R5LB     RESET ONE'S MINUTE DIGIT TO ZERO
*------------------------------------------------------------------------
       A    R10,R5       INCREMENT TEN'S MINUTES DIGIT
       CB   @R8LB,R5     SEE IF 60 MINUTES HAVE PASSED
       JGT  SCRN         NO!
       MOVB R8,R5        RESET TEN'S MINUTE DIGIT
*------------------------------------------------------------------------
       A    R10,R4       INCREMENT ONE'S HOUR DIGIT
       CB   @R7LB,@R3LB  TEN'S HOUR DIGIT A 0 OR 1?
       JGT  CLK1         YES!
       CB   @R9LB,R4     IS THE ONE'S HOUR DIGIT PAST 3?
       JGT  CLK1         NO!
       MOVB R9,R4        SET ONE'S HOUR DIGIT BEFORE ENTERING CLK1
*------------------------------------------------------------------------
CLK1   CB   R9,R4        HAS 12 HOURS PASSED YET?
       JGT  SCRN         NO!
       MOVB R8,R4        CLEAR ONE'S HOUR DIGIT
       INC  R3           INCREMENT TEN'S HOUR DIGIT
       CB   @R7LB,@R3LB  HAVE 24 HOURS PASSED?
       JHE  SCRN         NO!
       MOVB R8,@R3LB     CLEAR TEN'S HOUR DIGIT
SCRN   LI   R0,22        LOAD SCREEN LOCATION FOR CLOCK
       LI   R1,WSR+6     LOAD DATA LOCATION TO MOVE
       LI   R2,9         LOAD NUMBER OF BYTES TO MOVE
       BLWP @VMBW        MOVE THE CLOCK TO THE SCREEN
       LWPI GPLWS        RELOAD GPL WORKSPACE
       RT                RETURN TO INTERRUPT ROUTINE
*========================================================================

* SETCLOCK ROUTINE

SETCLK LWPI WSR          LOAD OUR WORKSPACE REGISTERS
       LI   R12,>0006    MAKE SURE R12 HAS A 6 IN THE MSB FOR THE STRREF ROUTINE
       LI   R0,TIMER     LOAD R0 WITH 60
       MOV  R0,@COUNTR   RELOAD OUR MINUTE TIMER
       CLR  R0           WE WON'T BE LOADING AN ARRAY SO CLEAR R0
       LI   R1,1         ONLY ONE PARAMETER BEING PASSED TO US THROUGH CALL LINK
LI   R2,R12LB      WHERE TO PUT THE TIME STRING
BLWP @STRREF       GET THE TIME STRING
MOVB R13,@R3LB     MOVE THE TEN'S HOUR DIGIT INTO PLACE
MOVB @R13LB,R4     MOVE THE ONE'S HOUR DIGIT INTO PLACE
MOV  R14,R5        MOVE THE ONE'S AND TEN'S MINUTE DIGITS INTO PLACE
MOVB R15,@R6LB     MOVE THE TEN'S SECOND DIGIT INTO PLACE
MOVB @R15LB,R7     MOVE THE ONE'S SECOND DIGIT INTO PLACE
AB   R11,@R3LB     ADD THE BASIC SCREEN OFFSET TO OUR TIME
AB   R11,R4              .
A    R11,R5              .
AB   R11,@R6LB           .
AB   R11,R7              .
MOVB R0,@STATUS    CLEAR THE STATUS BYTE
LWPI GPLWS         LOAD THE GPL WORKSPACE REGISTERS
B    @NEXT         RETURN TO THE GPL INTERPRETER (i.e. BASIC IN OUR CASE)
END
```

Here is the CALL LOAD Version of the CLOCK and SETCLK Assembly routines.

```
10 CALL CLEAR :: CALL INIT

20 CALL LOAD(16368,83,69,84,
67,76,75,37,152)

30 CALL LOAD(16376,67,76,79,
67,75,32,37,24)

40 CALL LOAD(8194,37,226,63,
240)

50 CALL LOAD(9460,0,60,0,0,0
,0,0,0,128,144,144,154,144,1
44,154,144,144,146,144,150,1
54,148)

60 CALL LOAD(9482,1,0,96,96,
0,6,205,75,205,96,33,131,37,
30,200,32,37,22,131,196,2,22
4)

70 CALL LOAD(9504,36,246,6,3
2,36,244,22,45,2,0,0,60,200,
0,36,244,161,202,145,201,21,
38)

80 CALL LOAD(9526,209,200,5,
134,152,32,37,7,37,3,21,32,2
16,8,37,3,5,133,152,9,37,1)

90 CALL LOAD(9548,21,26,216,
8,37,1,161,74,145,96,37,7,21
,20,209,72,161,10,152,32,37,
5)

100 CALL LOAD(9570,36,253,21
,4,145,32,37,9,21,1,209,9,14
5,9,21,8,209,8,5,131,152,32)

110 CALL LOAD(9592,37,5,36,2
53,20,2,216,8,36,253,2,0,0,2
2,2,1,36,252,2,2,0,9)

120 CALL LOAD(9614,4,32,32,3
6,2,224,131,224,4,91,2,224,3
6,246,2,12,0,6,2,0,0,60)

130 CALL LOAD(9636,200,0,36,
244,4,192,2,1,0,1,2,2,37,15,
4,32,32,20,216,13,36,253)

140 CALL LOAD(9658,209,32,37
,17,193,78,216,15,37,3,209,2
24,37,21,184,11,36,253,177,1
1,161,75)

150 CALL LOAD(9680,184,11,37
,3,177,203,216,0,131,124,2,2
24,131,224,4,96,0,112,88,79)

160 CALL LINK("CLOCK")

170 DISPLAY AT(12,1):"TIME?_
______" :: ACCEPT AT(12,6)SIZ
E(-6)BEEP VALIDATE(DIGIT):TI
ME$ :: CALL LINK("SETCLK",TI
ME$)

180 CALL CLEAR :: CALL LOAD(
-31952,255,231,255,231)!
   This is like 'NEW'
```

## RS232 CARD - DSR MEMORY MAP

```
+------------------------------------------------------------------------------+
|        | RS232 DSR HEADER                                                    |
|        |  Note: This header MUST be at >4000 for a valid DSR                 |
|  >4000 |  >AA01  Header Validation byte and Version Number                   |
|  >4002 |  >0000  Number of Application Programs - not used in DSRs           |
|  >4004 |  >4010  Power Up Header Address                                     |
|  >4006 |  >0000  Application Program Header - only used in cart GROM/ROM     |
|  >4008 |  >4016  DSR Header Address                                          |
|  >400A |  >0000  Subprogram Header Address - none here                       |
|  >400C |  >406C  Interrupt Link Header Address                               |
|  >400E |  >0000  Not Used - Reserved                                         |
|        |                                                                     |
|        | POWER UP HEADER                                                     |
|  >4010 |  >0000  Link to next Power Up Header - no more                      |
|  >4012 |  >40F4  Entry Point for this Power Up Routine                       |
|  >4014 |  >0000  Not Used                                                    |
|        |                                                                     |
|        | DSR HEADER(s)                                                       |
|  >4016 |  >4020  Link to next DSR Header                                     |
|  >4018 |  >416E  Entry Point for this DSR Routine                            |
|  >401A |    >05  Length of DSR Routine Name                                  |
|  >401B |  >5253323332 RS232  DSR Link Name                                   |
|        |                                                                     |
|  >4020 |  >402C  Link to next DSR Header                                     |
|  >4022 |  >416E  Entry Point for this DSR Routine                            |
|  >4024 |    >07  Length of DSR Routine Name                                  |
|  >4025 |  >52533233322F31 RS232/1  DSR Link Name                             |
|        |                                                                     |
|  >402C |  >4038  Link to next DSR Header                                     |
|  >402E |  >4174  Entry Point for this DSR Routine                            |
|  >4030 |    >07  Length of DSR Routine Name                                  |
|  >4031 |  >52533233322F32 RS232/2  DSR Link Name                             |
|        |                                                                     |
|  >4038 |  >4040  Link to next DSR Header                                     |
|  >403A |  >415E  Entry Point for this DSR Routine                            |
|  >403C |    >03  Length of DSR Routine Name                                  |
|  >403D |  >50494F PIO  DSR Link Name                                         |
|        |                                                                     |
|  >4040 |  >404A  Link to next DSR Header                                     |
|  >4042 |  >415E  Entry Point for this DSR Routine                            |
|  >4044 |    >05  Length of DSR Routine Name                                  |
|  >4045 |  >50494F2F31 PIO/1  DSR Link Name                                   |
|        |                                                                     |
|  >404A |  >4054  Link to next DSR Header                                     |
|  >404C |  >4164  Entry Point for this DSR Routine                            |
|  >404E |    >05  Length of DSR Routine Name                                  |
|  >404F |  >50494F2F32 PIO/2  DSR Link Name  (for second RS232 Card)          |
|        |                                                                     |
|  >4054 |  >4060  Link to next DSR Header                                     |
|  >4056 |  >4180  Entry Point for this DSR Routine                            |
|  >4058 |    >07  Length of DSR Routine Name                                  |
|  >4059 |  >52533233322F33 RS232/3  DSR Link Name  (for second RS232 Card)    |
|        |                                                                     |
|  >4060 |  >0000  Link to next DSR Header - no more                           |
|  >4062 |  >417A  Entry Point for this DSR Routine                            |
|  >4064 |    >07  Length of DSR Routine Name                                  |
|  >4065 |  >52533233322F34 RS232/4  DSR Link Name  (for second RS232 Card)    |
+------------------------------------------------------------------------------+
```

## RS232 CARD - DSR MEMORY MAP Continued

| | |
|---|---|
| | **INTERRUPT HEADER** |
| >406C | >0000 Link to next Interrupt Header - no more |
| >406E | >40D2 Entry Point for Interrupt Link Routine |
| >4070 | >0000 Not Used - Reserved |
| | |
| >4072 | >0800 Data for Eight & Zero |
| >4074 | >0303 Data for Three |
| | **OPTIONS TABLE** |
| >4076 | >4543 **EC** >4512 Entry Point for EC. |
| >407A | >4352 **CR** >4518 Entry Point for CR. |
| >407E | >4C46 **LF** >451E Entry Point for LF. |
| >4082 | >4E55 **NU** >4524 Entry Point for NU. |
| >4086 | >4441 **DA** >4570 Entry Point for DA. |
| >408A | >4241 **BA** >4536 Entry Point for BA. |
| >408E | >5041 **PA** >4540 Entry Point for PA. |
| >4092 | >5457 **TW** >4596 Entry Point for TW. |
| >4096 | >4348 **CH** >452A Entry Point for CH. |
| >409A | >0000 End of Options Table |
| | |
| | **CLOCK VALUE POINTER TABLE** |
| >409C | >0028 Value at >000C in Console ROM for 2.5 MHz Clock |
| >409E | >40B6 Address of Values for 2.5 MHz Baud Rates |
| >40A0 | >0030 Value at >000C in Console ROM for 3.0 MHz Clock |
| >40A2 | >40C4 Address of Values for 3.0 MHz Baud Rates |
| >40A4 | >0000 End of Table |
| | **BAUD RATE TABLE** |
| >40A6 | >006E 110 Baud |
| >40A8 | >012C 300 Baud |
| >40AA | >0258 600 Baud |
| >40AC | >04B0 1200 Baud |
| >40AE | >0960 2400 Baud |
| >40B0 | >12C0 4800 Baud |
| >40B2 | >2580 9600 Baud |
| >40B4 | >0000 End of Table |
| | **2.5 MHz CLOCK VALUES** |
| >40B6 | >8563 >8482 >8209 >015B >8082 >8041 >002B |
| | **3.0 MHz CLOCK VALUES** |
| >40C4 | >85AA >849C >8271 >01A1 >809C >804E >8027 |
| | |
| >40D2 | **INTERRUPT ROUTINE ENTRY POINT**<br>This is the start of the Interrupt driven Circular Input Buffer Routine. This routine allows RS232, RS232/1 & RS232/2 inputs on interrupts and places the data in a predetermined VDP Buffer. |
| | |
| >40F4 | **POWER UP ROUTINE**<br>This is executed when the computer first powers up. This routine initializes the 9901 (PIO) and 9902s (RS232/1 & /2). |
| | |
| >401E | This is the balance of the Interrupt handling routine. |
| | |
| >415E | PIO & PIO/1 Entry Point |
| >4164 | PIO/2 Entry Point (for second RS232 Card) |
| >416E | RS232 & RS232/1 Entry Point |
| >4174 | RS232/2 Entry Point |
| >417A | RS232/4 Entry Point (for second RS232 Card) |
| >4180 | RS232/3 Entry Point (for second RS232 Card) |

## RS232 CARD - DSR MEMORY MAP Continued

| | |
|---|---|
| >418E | ALL RS232s JUMP TO HERE |
| >4190 | ALL PIOs JUMP TO HERE and then they jump to one of the routines in the following table depending on the operation requested |
| | |
| | **OPCODE VECTOR TABLE** |
| >4202 | >4210 Entry Point for OPEN |
| >4204 | >4464 Entry Point for CLOSE |
| >4206 | >4236 Entry Point for READ |
| >4208 | >42FA Entry Point for WRITE |
| >420A | >4450 Entry Point for RESTORE/REWIND Illegal Opcode |
| >420C | >4338 Entry Point for LOAD |
| >420E | >43D2 Entry Point for SAVE |
| | |
| >4210 | **OPEN** Routine (OPEN #1:"RS232".BA=xxxx etc.) |
| >4236 | **READ** Routine (INPUT #1: A$) |
| >42FA | **WRITE** Routine (PRINT #1: A$) |
| >4338 | **LOAD** Routine (OLD RS232) |
| >43D2 | **SAVE** Routine (SAVE RS232) |
| >444A | Error Handling Routines |
| >4450 | Illegal Opcode Handler |
| >4464 | **CLOSE** Routine (CLOSE #1) |
| | |
| >4490 | Routine to parse for the OPTIONS (BA, CR LF etc.) |
| | |
| | **ROUTINES TO SET UP THE OPTIONS** |
| >4512 | ECHO - EC Option Routine (.EC) |
| >4518 | CARRIAGE RETURN - CR Option Routine (.CR) |
| >451E | LINE FEED - LF Option routine (.LF) |
| >4524 | NULL - NU Option Routine (.NU) |
| >452A | CHECK PARITY - CH Option Routine (.CH) |
| >4536 | BAUD RATE - BA Option Routine (.BA=1200) |
| >4540 | PARITY - PA Option Routine (.PA=E) |
| >4570 | DATA BITS - DA Option Routine (.DA=8) |
| >4596 | TWO STOP BITS - TW Option Routine (.TW) |
| | |
| | **MISC SUBROUTINES AND ROUTINES** |
| >45A0 | The routines in this area are used by the above routines to set up registers and for parsing inputted values for Baud Rate etc. |
| >463A | This routine reads a single character in from an RS232 port. |
| >466A | This routine reads a single character in from the PIO port. |
| >4686 | This routine places the block counter values on the screen during a LOAD or SAVE .... Opcode |
| >46EE | This routine sends the Carriage Return character when needed. |
| >4700 | This routine sends the NULL characters when needed. |
| >4740 | This routine checks for an INTERNAL Data type. |
| >474A | This routine checks for a FIXED Record Length. |
| >4754 | This routine converts ASCII Values into Binary. |
| >4798 | This is the Scan routine. It finds non-space characters. |
| >47E4 | This routine Transmits a character to an RS232 port. |
| >4808 | This routine Transmits a character to the PIO port. |
| >4822 | These routines set up the RS232 ports and PIO port, clear the VDP Screen area for the SAVE and LOAD value, check Ready to Read check Status and cause a time delay between PIO characters. |
| >5000 | The 1 Byte buffer used by the RS232 Card (not fully decoded) |

## ASSEMBLY LANGUAGE PRINT ROUTINES

by Edgar Dohmann -- JSC User's Group (JUG)

Here are some general print routines which I have developed for use in assembly language programs. These routines are set up as BLWP subroutines to isolate their register usage from your main assembly language program which calls them. Included in the listings is a PAB definition for my printer ("RS232.BA=600"). Substitute your printer's description in PNAME and be sure to change the value of PNAML to reflect the length in bytes of your printer's description.

The PAB locations used here are >1F00 for the PAB description and >1F40 for the line buffer to be printed. You may use other areas of VDP for your PAB and buffer if you like, but make sure they are not being used by the computer for something else.

```
REF VSBW,VMBW,DSRLNK

PABLOC EQU >1F00 VDP LOCATION OF PAB
DATLOC EQU >1F40 VDP LOCATION OF LINEP
PABPNT EQU >8356 POINTER TO PAB

DATBUF BSS 80        80-BYTE LINE BUFFER
PRTWSP BSS 32        WORKSPACE FOR ROUTINES

* **PAB DEFINITION**
PPAB    DATA >0012   OPEN CODE & FLAGS
        DATA DATLOC  LOCATION OF BUFFER
        DATA >5050   RECORD LENGTH
        DATA 0
PNAML   DATA 12      LENGTH OF PRINTER NAME
PNAME   TEXT 'RS232.BA=600' PRINTER NAME
PPABE   EQU $        END OF PAB DEFINITION

PCLS    BYTE 1       CLOSE CODE
PWRT    BYTE 3       WRITE CODE

POPEN   DATA PRTWSP  BLWP VECTOR FOR OPEN
        DATA POPN
PCLOS   DATA PRTWSP  BLWP VECTOR FOR CLOSE
        DATA PCLO
POUTP   DATA PRTWSP  BLWP VECTOR FOR OUTPUT
        DATA POUT

POPN    LI   R0,PABLOC    GET VDP ADDRESS
        LI   R1,PPAB      POINT TO PAB DEF
        LI   R2,PPABE-PPAB  LENGTH OF PAB
        BLWP @VMBW        MOVE PAB TO VDP
        LI   R6,PABLOC+9 ADDRESS TO SAVE
        MOV  R6,@PABPNT     IN PAB POINTER
        BLWP @DSRLNK      OPEN PRINTER
        DATA 8
        LI   R0,PABLOC    GET VDP ADDRESS
        MOVB @PWRT,R1     SET FOR WRITE
        BLWP @VSBW          IN PAB
        RTWP

PCLO    LI   R0,PABLOC    GET VDP ADDRESS
        MOV  @PCLS,R1     SET FOR CLOSE
        BLWP @VSBW          IN PAB
        LI   R6,PABLOC+9 ADDRESS TO SAVE
        MOV  R6,@PABPNT     IN PAB POINTER
        BLWP @DSRLNK      CLOSE PRINTER
        DATA 8
        RTWP

POUT    LI   R0,DATLOC    VDP ADDR OF BUFFER
        LI   R1,DATBUF    POINT TO BUFFER
        LI   R2,80        80-BYTE LINE BUFFER
        BLWP @VMBW        MOVE LINE TO VDP
        LI   R6,PABLOC+9 ADDRESS TO SAVE
        MOV  R6,@PABPNT     IN PAB POINTER
        BLWP @DSRLNK      WRITE A LINE
        DATA 8
        RTWP
```

Here is a program that can be used to test the print routines given above. The DEF statement will cause the program to be included in the REF/DEF table when it is loaded. The assembled object code can be loaded by either the LOAD AND RUN option of the Editor/Assembler or by a CALL LOAD from Basic and Extended Basic. If either Basic is used, a CALL LINK will have to follow the load to execute the program.

The test program given here will print two lines over and over until you reset the computer. For convenience, two additional routines are included with the program: PCLEAR will clear the line buffer in RAM and MOVMSG will copy a message into the line buffer to prepare it for printing.

```
*             **ROUTINE FOR TESTING**
       DEF  TEST
TSTWSP BSS  32            MY WORKSPACE
TEST   LWPI TSTWSP
       BLWP @POPEN        OPEN PRINTER
       BL   @PCLEAR       CLEAR BUFFER
       LI   R0,MESG1      MESSAGE TO PRINT
       LI   R1,MESG1E-MESG1 LENGTH OF MESSG
       BL   @MOVMSG       MOVE TO LINE BUFF
       BLWP @POUTP        PRINT MESSAGE
       BL   @PCLEAR       CLEAR BUFFER
       LI   R0,MESG2      NEXT MESSAGE
       LI   R1,MESG2E-MESG2  LENGTH
       BL   @MOVMSG            MOVE IT
       BLWP @POUTP             PRINT IT
       BLWP @PCLOS        CLOSE PRINTER
       JMP  TEST          **LOOP BACK**
```

```
MESG1  TEXT 'TEST MESSAGE'
MESG1E EQU  $
MESG2  TEXT 'ANOTHER MESSAGE'
MESG2E EQU  $

PCLEAR LI   R0,>2020    LOAD 2 BLANKS
       LI   R1,40       80 BYTES = 40 WORDS
       LI   R2,DATBUF   LOCATION OF BUFFER

PCLR1  MOV  R0,*R2+     BLANK 2 BYTES
       DEC  R1          DONE 40 WORDS YET?
       JNE  PCLR1       LOOP TIL DONE
       RT
MOVMSG LI   R2,DATBUF   LOCATION OF BUFFER
MOVM1  MOVB *R0+,*R2+   MOVE A BYTE
       DEC  R1          MESSAGE MOVED?
       JNE  MOVM1       LOOP TIL DONE
       RT
```

As I mentioned above, you can load the program with either Basic or Extended Basic. However, as you may know, Extended Basic does not include a DSRLNK to allow programs like this to access peripheral devices. Fortunately there are several versions of DSRLNK floating around which you can include in your program if you have access to them. John Phillips, John Clulow, and I have each provided versions of DSRLNK to User's Groups through the 99'ers Users Group Association.

Another alternative is to use a pseudo-DSRLNK routine like the one below. This is a "stripped down" version of DSRLNK which is only good for one peripheral. The standard DSRLNK searches through all DSR ROMs until it finds a device name which matches the one specified in your PAB. This version only searches one ROM and is set up here to check the 1st RS232 card (CRU address of >1300).

This routine is intended for calling with a BL so the BLWP @DSRLNK calls in the printer routines above should be replaced with BL @DSRLK calls. Also the DATA 8 instructions following the BLWP calls must be deleted. The advantage of this shortened version is that it is less than half the size of the standard DSRLNK so it is easier to type, takes up less memory, loads faster, and executes faster. One other change that must be made is to delete the REF statement for VSBW, VMBW, and DSRLNK. The Extended Basic loader does not resolve REFerences and the routines VSBW and VMBW must be explicitly EQUated to their X-Basic values as follows:

```
                 VSBW EQU  >2020
                 VMBW EQU  >2024
```

The value of PNAMP must be matched to the name length of your printer but must only reflect the characters up to the first period of the name. For a printer description of PIO.LF set PNAMP to 3.

```
*        <<<    PSEUDO DSRLNK    >>>
PNAMP  EQU  5        LENGTH OF 'RS232'

DSRLK  LWPI >83E0      GPL WORKSPACE
       LI   R0,PNAMP   GET NAME LENGTH
       MOV  R0,@>8354  SAVE FOR DSR USE
       INC  R0         ADJUST FOR .
       A    R0,@>8356  ADJUST PAB POINTER
       LI   R12,>1300  CRU FOR 1ST RS232
       LI   R1,1       DSR VERSION #
       SBO  0          TURN ON DSR
       LI   R2,>4008   STD ADDR FOR DSR LINK
       JMP  SGO2
SGO    MOV  R3,R2      TRY NEXT DEVICE
SGO2   MOV  *R2,R2     GET NEXT LINK ADDR
       JEQ  NOROM      EXIT IF NO MORE
       MOV  R2,R3      SAVE LINKAGE
       INCT R2         POINT DSR FOR DEVICE
       MOV  *R2+,R9    SAVE IT MIGHT NEED IT
       LI   R5,PNAMP*>100   NAME LEN IN MSB
       CB   R5,*R2+    SEE IF LENGTH MATCHES
       JNE  SGO        NO
       SRL  R5,8       YES
       LI   R6,PNAME   GET ADDRESS OF NAME
NAME1  CB   *R6+,*R2+  SEE IF NAMES MATCH
       JNE  SGO        NO
       DEC  R5         YES
       JNE  NAME1
       BL   *R9        NAME MATCHES
       NOP             NEED ERROR RTN SPOT
       SBZ  0          TURN OFF DSR
NOROM  LWPI PRTWSP     PREPARE TO RETURN
       RT
```

One last point to mention is the fact that the CLOSE operation is not required for the RS232 peripheral. The PCLOS subroutine is included here mainly for completeness. Basic programs require the CLOSE operation to reclaim the VDP buffer space that was allocated when the "file" for the RS232 card was opened. However this is not necessary in assembly language and the DSR itself takes no action in response to a CLOSE command. CLOSE commands are required for real file oriented devices like disks because this causes the sector buffer in memory to be written to the disk (on write operations) and also causes the file directory (which is kept in memory while the file is open) to be written to the disk. Such activities are not necessary for devices like the RS232 card.

```
SCR #100
   ;S      CHIP'S SOUND ROUTINES... DOCUMENTATION
    These screens will allow you to use sound statements similar
 to those of TI-Basic in TI-Forth. An example which you may find
useful is also included.
    To use sound with the TI99/4A, you must first build a
sound list in VDP ram. The words SOUNDBUILD and NOISEBUILD will
help you do this. The format for SOUNDBUILD is:
  generator# frequency volume SOUNDBUILD. Note that you must
specify tone generator 1, 2, or 3 for SOUNDBUILD. The frequency
is in hertz, and must be a number between 110 and 32767.
The volume is a number between 0 and 30, 0 being the loudest, an
d 30 being the softest.(silence)  The format for NOISEBUILD is:
  noise-type volume NOISEBUILD. The noise type is a number from
0 to 7. If noise #7 is specified, the noise shift rate depends
on the frequency you specify for sound generator #3.

SCR #101
   ;S      CHIP'S SOUND ROUTINES  cont.
    After executing a series of SOUNDBUILDs and/or a NOISEBUILD,
you execute the word DURATION. The format for DURATION is :
   duration(in milliseconds)   DURATION.
The duration can be from 0 to 4095 milliseconds.
( Actually, the specification isn't exact--if you specify 4095
milliseconds, the sound will play for 4.25 seconds)
   When you've finished creating a sound list with a series of
SOUNDBUILDs, NOISEBUILDs, and DURATIONs, you can hear the list
you've built by executing the word PLAY. If you want to hear
your sound list again, use the word REPLAY. Here is an example:
After loading the screens which contain the sound routines, type
in the following:
1 330 0 SOUNDBUILD   2 440 5 SOUNDBUILD   <cr>
  3000     DURATION     <cr>    PLAY <cr>
  and then...REPLAY <cr>  if you wish to hear the sounds again
SCR #102
   ;S    CHIP'S SOUND ROUTINES last docs.
    The third screen of the sound routines shows another example
of the use of SOUNDBUILD and DURATION.
    Words which define musical notes start on the fourth screen.
The format is : generator# octave# NOTENAME
where NOTENAME is one of the notes which is defined in the
fourth screen. For example 1 2 @A 1000 DURATION PLAY would make
generator #1 play the note A natural for 1 second. #A refers to
A sharp and $A refers to A flat. See the screens following the
fourth one for an example of how to use the words on the fourth
screen. (These screens contain a TI99/4A rendition of the theme
to the TV series Star Trek.)
    I hope you enjoy using these screens. If you have any
questions, I'd be glad to answer them. My CompuServe user ID# is
74206,3252 and my name is Chip Jarvis. I can also be reached c/o
San Diego TI-SIG.  4013 HONEYCUTT ST. SAN DIEGO, CA 92109
SCR #103
( TESTED SOUND ROUTINES 2/14/85  CHIP JARVIS) BASE->R : SOUNDS ;
 HEX 1400 VARIABLE S-START   1400 VARIABLE L-DUR
     1401 VARIABLE S-END

: VDP-WRITE ( ...bytes to write, # of bytes)
    0 DO S-END @ VSBW 1 S-END +! LOOP ;
: VOICE 1- 20 * 80 OR ;  DECIMAL
: FREQ-CODE ( gen freq -- )
    111861. ROT M/ SWAP DROP DUP 4 SRA ROT VOICE ROT 15 AND OR
    2 VDP-WRITE ;
: VOL-CODE ( gen vol -- )
    1 SRA SWAP VOICE 16 + OR 1 VDP-WRITE ;
: NOISE-CODE ( noisetype -- )
    7 AND 224 OR 1 VDP-WRITE ;
: DUR-CODE ( millisec -- 60ths)   4 SRA 1 VDP-WRITE ;
                                                        -->
```

```
SCR #104
 ( SOUND ROUTINES CONT. )
 : UPDATE-POINTERS ( -- )
   S-END @ L-DUR @ - DUP 1-  L-DUR @ VSBW 1+ L-DUR +! ;

 : SOUNDBUILD ( gen freq vol --)
   ROT DUP ROT VOL-CODE SWAP FREQ-CODE ;
 : NOISEBUILD ( type vol --)
   4 DUP ROT VOL-CODE SWAP NOISE-CODE ;
 : DURATION ( millisec -- )
   UPDATE-POINTERS DUR-CODE S-END 1 +! ;
        HEX
 : REPLAY ( replay sound list in vdp ram)
       S-START @ 83CC !   83FD C@ 1 OR 83FD C!    1 83CE C! ;
 : PLAY 00 FF DF BF 9F 04   6 VDP-WRITE
       1401 S-END !  1400 L-DUR ! REPLAY ;
    R->BASE                                           -->
SCR #105
  ( CHARGE! 2/14/85 T. CHIP JARVIS)
    1 1047 0 SOUNDBUILD 125 DURATION
    1 1175 0 SOUNDBUILD 125 DURATION
    1 1319 0 SOUNDBUILD 125 DURATION
    1 1568 0 SOUNDBUILD 125 DURATION
    1 30000 0 SOUNDBUILD 125 DURATION
    1 1319 0 SOUNDBUILD 175 DURATION
    1 1568 0 SOUNDBUILD 600 DURATION
  PLAY ;S







SCR #106
  ( NOTE DEFINITIONS 2/16/85 C.JARVIS)  1000 VARIABLE FULLCOUNT
  : NOTE <BUILDS , DOES> @ 2 / SWAP 0 DO 2 * LOOP 0 SOUNDBUILD ;
  : NOTELEN <BUILDS , DOES> @ FULLCOUNT @ SWAP / DURATION ;
  : NOTELEN. <BUILDS , DOES> @ FULLCOUNT @
                           DUP 2 / + SWAP / DURATION ;
  110 NOTE @A
  116 NOTE #A    : $B #A ;       16 NOTELEN. SIXT.
  123 NOTE @B                    16 NOTELEN  SIX
  131 NOTE @C    : $C @B ;        8 NOTELEN. EIGT.
  139 NOTE #C                     8 NOTELEN  EIGT
  147 NOTE @D    : $D #C ;        4 NOTELEN. QUAR.
  156 NOTE #D    : $E #D ;        4 NOTELEN  QUAR
  165 NOTE @E    : $F @E ;        2 NOTELEN. HALF.
  175 NOTE @F                     2 NOTELEN  HALF
  185 NOTE #F    : $G #F ;        1 NOTELEN. WHOL.
  196 NOTE @G   208 NOTE #G     : $A 1 - #G ;     1 NOTELEN  WHOL
SCR #107
   ( STAR TREK MUSIC 2/16/85 CHIP JARVIS)
   : *B @B ;  : @B #A ; ( B's flat) : CE 1 2 @C 2 2 @E ;
  1 2 @C QUAR                    1 2 @C 2 1 @C          QUAR
  1 2 @C  2 3 @B 3 1 @F QUAR     1 2 @C 2 3 @B 3 1 @C QUAR
  1 2 @C  2 3 @B        QUAR     1 3 @A 2 1 @C          QUAR
  1 2 @G  2 2 @A 3 1 @F EIGT.    1 2 @F 2 2 @A 3 1 @F EIGT.
      1 2 @E 2 2 @A 3 1 @F EIGT. 1 2 *B 2 2 #D          HALF
  CE             3 1 @F QUAR     CE               3 1 @A QUAR

  CE                    QUAR     CE               3 1 @E QUAR
  CE             3 1 #D QUAR     CE               3 1 #C QUAR
  1 2 @C                QUAR     1 2 @C 2 1 @C          QUAR
  1 3 @C 2 3 @A 3 2 @A  QUAR     1 3 @C 2 3 @A 3 1 @A QUAR
  1 3 @C 2 3 @A         QUAR     1 3 @B 2 1 @C          QUAR
  1 3 @A 2 2 @A 3 1 @F  EIGT.    1 2 @G 2 2 @A 3 1 @F EIGT.
      1 2 @F 2 2 @A 3 1 @F EIGT.                          -->
```

```
SCR #108
( STAR TREK cont. 2/16/85 TCJ)
CE                    QUAR     CE   3 1 @D             QUAR
CE  3 1 #G            QUAR     CE   3 1 @D             QUAR
CE                    QUAR     CE   3 1 @D             QUAR
CE  3 1 #G            QUAR     1 2 #D 2 1 @D           QUAR
1 2 *B 2 2 @D         QUAR     1 2 *B 2 2 @D  3 1 @D   QUAR
1 2 *B 2 2 @D 3 1 @G  QUAR     1 2 @E 2 2 @D           QUAR

1 2 #F                QUAR     1 2 @G 2 1 @G           QUAR
1 3 @A 2 1 @D         QUAR   1 3 *B EIGT   1 3 @C EIGT
1 2 @E 2 2 #G 3 3 @D  HALF
1 2 @F 2 3 @A 3 3 #D  WHOL.



                                                        -->
```

```
SCR #109
( STAR TREK cont. 2/16/85 TCJ)
1 2 @C 2 2 #D         QUAR     1 2 @C 2 2 #D 3 1 #D QUAR
1 2 @C 2 2 #D 3 1 #G  QUAR     1 2 @F 2 1 @E        QUAR
1 2 @C 2 2 @G         QUAR     1 2 #G 2 1 #D        QUAR
1 3 @B 2 2 @C 3 1 #G  EIGT.    1 3 @C 2 2 @C 3 1 #G EIGT.
   1 3 #C 2 2 @C 3 1 #G EIGT.
1 2 @E 2 2 #G 3 3 #D  HALF     1 2 @G 2 3 #A 3 1 @E HALF

1 2 @G 2 3 #A 3 1 @E  WHOL
1 2 @C                QUAR     1 2 @C 2 1 @C        QUAR
1 2 @C 2 3 @B 3 1 @F  QUAR     1 2 @C 2 3 @B 3 1 @C QUAR
1 2 @C 2 3 @B         QUAR     1 3 @A 2 1 @C        QUAR
1 2 @G 2 2 @A 3 1 @F  EIGT.    1 2 @F 2 2 @A 3 1 @F EIGT.
   1 2 @E 2 2 @A 3 1 @F EIGT.


                                                        -->
```

```
SCR #110
( STAR TREK cont. 2/16/85 TCJ)
1 2 *B 2 2 #D         HALF
1 2 @C 2 2 @E 3 1 @F  QUAR     1 2 @C 2 2 @E 3 1 @A QUAR
1 2 @C 2 2 @E         QUAR     1 2 @C 2 2 @E 3 1 @F QUAR
1 2 @C 2 2 @E 3 1 #D  QUAR     1 2 #C 2 1 #C        QUAR
1 2 @C                QUAR     1 2 @C 2 1 @C        QUAR
1 3 @A 2 3 @C 3 2 @A  QUAR     1 3 @A 2 3 @C 3 1 @C QUAR

1 3 @A 2 3 @C         QUAR     1 3 @B 2 1 @C        QUAR
1 3 @A 2 2 @A 3 1 @F  EIGT.    1 2 @G 2 2 @A 3 1 @F EIGT.
     1 2 @F 2 2 @A 3 1 @F EIGT.
1 2 @E 2 2 @D 3 1 #G  HALF
1 2 @E        3 1 #G  HALF
1 2 @E 2 2 #C 3 1 @G  HALF.
1 2 #D        3 1 @G  QUAR
                                                        -->
```

```
SCR #111
( STAR TREK cont. 2/16/85 T. CHIP JARVIS)
1 2 @B 2 2 @D 3 1 @B HALF
1 2 @B 2 2 @D        QUAR     1 2 @E               QUAR
1 2 #C 2 2 @F        QUAR     1 2 #C 2 2 @G 3 1 @B QUAR
1 3 @A 2 2 #C 3 1 #D EIGT.    1 3 @B 2 2 #C 3 1 #D EIGT.
    1 3 @A 2 2 #C 3 1 #D EIGT.
1 3 @A 2 3 @C        QUAR     1 3 @A 2 3 @C 3 1 @C QUAR
1 3 @A 2 3 @C 3 2 @A QUAR     1 3 @A 2 3 @C 3 1 @C QUAR

1 2 #F 2 3 #A 3 3 #D HALF.    1 2 #F 2 3 #A 3 3 @D QUAR
1 2 @F 2 3 @C        QUAR     1 2 @F 2 3 @C 3 1 @D QUAR
1 2 @D 2 2 @F 3 2 @B QUAR     1 2 @D 2 2 @F 3 1 @D QUAR
1 2 @D 2 2 @F 3 1 @C HALF
1 2 @E 2 2 @G 3 2 @B EIGT.    1 2 @E 2 9 @A 3 2 @B SIX
1 2 @E 2 2 @G 3 2 @B EIGT.    1 2 @E 2 9 @A 3 2 @B SIX
1 2 @E 2 2 @G 3 2 @B EIGT.    1 2 @E 2 9 @A 3 2 @B SIX   -->
```

```
SCR #112
  0   ( STAR TREK last 2/16/85 T. CHIP JARVIS)
  1   1 2 @A 2 2 @D 3 2 @F   WHOL
  2   1 2 @B 2 2 #D 3 2 @F   WHOL
  3   1 2 @C 2 3 @A          WHOL
  4   1 2 @C 2 3 @A          WHOL    ;S
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
```

Many thanks to Chip Jarvis. - We also received this method of generating sounds in TI Forth from Rex Nielsen - Thanks.

```
SCR #92
  0 ( Sound Routine - 3 Notes, 1 Noise Vol 1-16 by Rex Nielsen )
  1 BASE->R HEX
  2   80 VARIABLE OPER
  3 3001 VARIABLE SDTAB
  4    0 VARIABLE TIME   0 VARIABLE NOISY 2 ALLOT
  5 049F VARIABLE SDOFF  4 ALLOT BFDF SDOFF 2+ ! FF00 SDOFF 2+ 2+ !
  6 : OTEST  BEGIN 83CE @ 0 = UNTIL ;
  7 : +SDTAB 1 SDTAB +! ;
  8 : +OPER  10 OPER  +! ;
  9 : DUR    ABS 4 SRL DUP 1 < IF DROP 1 ENDIF TIME ! ;
 10 : TONE1  0 3000 VSBW 1B4F5. ROT U/ DUP F AND OPER @ +
 11          SDTAB @ VSBW +OPER +SDTAB 4 SRA 3F AND SDTAB @ VSBW
 12          +SDTAB DROP OPER @ + SDTAB @ VSBW +SDTAB +OPER ;
 13 : TONE2  TIME @ SDTAB @ VSBW SDTAB @ 3001 - 3000 VSBW +SDTAB
 14          SDOFF SDTAB @ 6 VMBW 3000 83CC ! 83FD DUP @ 01 OR
 15          SWAP ! 0100 83CE ! ;                    R->BASE  -->

SCR #93
  0 ( Sound Cont - Syntax =  0 VN -N V3 F3 V2 F2 V1 F1 DUR SOUND  )
  1 (       freq's F1 F2 F3 are like BASIC's                      )
  2 (       vol's  VN V1 V2 V3 are from 1 through 16              )
  3 (       dur is also like BASIC's - positive or negative       )
  4 (       you can use from 1 to 3 notes and/or 1 noise.         )
  5 (  note: first item on the stack MUST be a 0, zero, or the    )
  6 (        empty stack error message will come up.              )
  7 BASE->R HEX
  8
  9 : NOISE ABS 1 - 7 AND E0 + NOISY ! F0 + NOISY 2 + ! ;
 10 : T3 NOISY @ 0 > IF 3 0 DO NOISY I + @ SDTAB @ VSBW +SDTAB
 11      2 +LOOP ENDIF ;
 12 : -TEST DUP FFF7 U< IF TONE1 ELSE NOISE ENDIF ;
 13 : SOUND DUP 0 > IF OTEST ENDIF DUR BEGIN -TEST DUP 0 = UNTIL T3
 14         DROP TONE2 80 OPER ! 3001 SDTAB ! 0 NOISY ! ;
 15 R->BASE
```

Here are a few examples using the SOUND word on screens 92-93:

```
     FORTH                                    BASIC

0 1 110 100 SOUND <enter>               CALL SOUND(100,110,1)
0 2 220 3 330 4 440 100 SOUND <enter>   CALL SOUND(100,440,4,330,3,220,2)
0 14 -1 100 SOUND <enter>               CALL SOUND(100,-1,14)
0 3 -3 2 660 3 880 -200 SOUND <enter>   CALL SOUND(-200,880,3,660,2,-3,3)
```

# PC NOTES

We have two subprograms that save and restore the function key assignments in Basic and a program that generates Piecharts for you this issue.

The Piechart program requires the Three Planes Graphics board. This program allows you to title your chart and then input up to 100 values and names for each of the sections of the chart. We have found that charts with more than 20 items start to get a little cluttered so it is best to keep the number of pie sections limited. If you need to constantly generate piecharts with more than 20 items you might want to modify the program to label each section with a letter of the alphabet (A,B,C, etc.). Then modify the code to print the label and your section name down the sides of the screen. If you have the PRTSCN.DEV file installed (see the May issue) you can press ALT SHIFT PRINT and dump the piechart to your dot matrix printer.

The subprograms FKEY1 and FKEY2 should be saved in ASCII format so you can merge one of them into your other programs. FKEY1 has the proper address for Basic Version 1.1 and FKEY2 is set up for Basic Version 1.2. Both versions save the function keys into one 192 byte string. When the keys are restored with this subprogram you will have to execute a KEY ON to see them. You should place one of these in any of your programs that change the function keys and execute a GOSUB 65500 at the beginning of the program and a GOSUB 65520 at the end of the program. Have Fun.

**FKEY1 - MS-Basic Version 1.1**

```
65500 FOR I=1587 TO 1778: FKEY$=FKEY$+CHR$(PEEK(I)):    NEXT: RETURN  '    Save
65520 FOR I=1 TO 192: POKE I+1586,ASC(MID$(FKEY$,I,1)): NEXT: RETURN  ' Restore
```

**FKEY2 - MS-Basic Version 1.2**

```
65500 FOR I=2324 TO 2515: FKEY$=FKEY$+CHR$(PEEK(I)):    NEXT: RETURN  '    Save
65520 FOR I=1 TO 192: POKE I+2323,ASC(MID$(FKEY$,I,1)): NEXT: RETURN  ' Restore
```

**PIECHART - MS-Basic Version 1.1 & 1.2**

```
10  CLS: KEY OFF        '<<<  TI PC PIECHART  >>>  Version 2.0  CGM  8-20-84
20  COLOR 7,0,,0: LOCATE,,0: PRINT: DEFINT I: DEFDBL A-D,X,Y: DIM A(100),A$(100)
30  FOR I=1 TO 16: KEY(I) ON: NEXT:  KEY(14)OFF: KEY(15)OFF
40  LINE(40,40)-(680,140),2,BF:  LINE(44,43)-(676,137),7,BF
50  LINE(48,46)-(672,134),1,BF:  LINE(52,49)-(668,131),0,BF
60            LOCATE  6,22: PRINT " T E X A S   I N S T R U M E N T S";
70  COLOR 2: LOCATE  8,22: PRINT "        Professional Computer";
80  COLOR 7: LOCATE 10,22: PRINT "P I E C H A R T   G E N E R A T O R";
90  COLOR 1: LOCATE 14,22: PRINT "             Version 2.0";
100 COLOR 2: LOCATE 24,22: PRINT "  <  Press Any Key To Continue  >";
110 T$=INKEY$: IF T$="" GOTO 110 ELSE IF T$<>CHR$(27) GOTO 130
120 RUN "FILEMENU
130 T=0: C#=1D-38: CLS: COLOR 6: LINE(0,0)-(719,299),,B
140 LOCATE 2,1: INPUT "  Title of Piechart       : ",T$
150 INPUT "  Number of Items in Chart: ",N : PRINT: COLOR 4
160 FOR I=1 TO N:INPUT "     Numeric Value, Name  : ",A(I),A$(I): T=T+A(I): NEXT
170 FOR I=1 TO N: A#(I)=(A(I)/T)*6.28318530718#: NEXT: CLS: COLOR 6: T=LEN(T$)*9
180 LINE (0,17)-(719,299),,B: IF T THEN I=40-T/18: LOCATE 2,I: I=I*9  ELSE 200
190 LINE(I-28,17)-(T+I+9,17),0:  LINE (I-28,7)-(T+I+9,27),,B:  PRINT T$;
200 FOR I=1 TO N:  B#=C#:  C#=C#+A#(I):  D#=(B#+C#)/2:  T=(C#-B#>.079#)*6
210 X=360+COS(D#)*14.3: Y=150-SIN(D#)*7.914: CIRCLE(X,Y),150,7+T,-B#,-C#+.000001
220 IF T THEN  PAINT(X+COS(D#)*143,Y-SIN(D#)*79.14466),I MOD 6+1,1
230 T=LEN(A$(I)):  IF T THEN COLOR 7,0  ELSE 260
240 X=(X+COS(D#)*200-4*LEN(A$(I)))\9: Y=(Y-SIN(D#)*100)\12+1
250 LOCATE Y,X+1: PRINT A$(I): LINE (X*9,Y*12)-(X*9+9*LEN(A$(I)),Y*12),1
260 NEXT:  BEEP
270 A$=INKEY$: IF A$="" THEN 270 ELSE IF A$="N" OR A$="n" THEN 120
280 IF A$="Y" OR A$="y" THEN 130 ELSE BEEP: GOTO 270
```

## SUBSCRIPTION INFORMATION

**THE SMART PROGRAMMER** - a monthly 16+ page newsletter published by **MILLERS GRAPHICS**
U.S. 12.50 year - Foreign Surface Mail 16.00 year - Foreign Air Mail 26.00 year

Back issues are available. We can start your subscription with the FEB. 84 issue
To subscribe send a Check, Money Order or Cashiers Check, payable in U.S. currency

TO: **MILLERS GRAPHICS**
**1475 W. Cypress Ave.**
**San Dimas, CA 91773**

**THE SMART PROGRAMMER** is published by MILLERS GRAPHICS, 1475 W. Cypress Ave., San Dimas, CA 91773. Each separate contribution to this issue and the issue as a whole All mail directed to **THE SMART PROGRAMMER** will be treated as unconditionally assigned for publication and copyright purposes and is subject to **THE SMART PROGRAMMER'S** unrestricted right to edit and comment. MILLERS GRAPHICS assumes no liability for errors in articles.

MG
**MILLERS GRAPHICS**
1475 W. Cypress Ave.
San Dimas, CA 91773

*THE SMART PROGRAMMER*